EcomLabs

STEP-BY-STEP GUIDE

Shopify theme app extensions

The complete example extension, with an app block, an app embed, their assets and their translations; how to validate and ship it with Shopify CLI; how to retire script tags without a gap in service, and the checklist for cleaning up the code an app left in a theme.

FOR	Developers of Shopify apps that add content or scripts to a storefront
TESTED WITH	Shopify CLI 3.88.1 · shopify theme check · Node 22.22.3 · not connected to any store
UPDATED	November 19, 2026
DOCS	shopify.dev/docs/apps/build/online-store/theme-app-extensions
ARTICLE	ecomlabs.dev/g/extension-en
AUTHOR	Luis Morales · @luismoralesdev

Contents

Before you start	What you will build
Step 01	The app block
Step 02	The app embed
Step 03	Translations
Step 04	Validate, ship and retire script tags
When you finish	Theme cleanup checklist

BEFORE YOU START

What you will build

An extension with two blocks: **stock-badge**, an app block that warns on the product page when stock runs low, and **help-bubble**, an app embed with a floating help link and its own JavaScript. You need an app created with Shopify CLI 3.0 or later, a dev store and an Online Store 2.0 theme to test the app block. From the app folder:

```
shopify app generate extension --template theme_app_extension --name stock-badge
```

Delete the sample star rating block (it needs a `demo.avg_rating` metafield). `shopify.extension.toml` ends up like this:

```
name = "stock-badge"
type = "theme"
```

STEP 01

The app block

`blocks/stock-badge.liquid`: `"target": "content"` makes it an app block, `autofill` takes the section's product, and the `inventory_management` check stops it from reading units sold when inventory is not tracked.

```
{%- liquid
  assign badge_product = block.settings.product
  assign variant = badge_product.selected_or_first_available_variant
  assign quantity = variant.inventory_quantity
-%}
{%- if variant.inventory_management == 'shopify' and quantity > 0 and quantity <= block.settings.threshold -%}
  <p class="stock-badge" style="--stock-badge-color: {{ block.settings.color }}">
    {{ 'stock_badge.low_stock' | t: quantity: quantity }}
  </p>
{%- endif -%}

{% schema %}
{
  "name": "t:blocks.stock_badge.name",
  "target": "content",
  "stylesheet": "stock-badge.css",
  "enabled_on": { "templates": ["product"] },
  "settings": [
    { "type": "product", "id": "product", "label": "t:blocks.stock_badge.product", "autofill": true },
    { "type": "range", "id": "threshold", "label": "t:blocks.stock_badge.threshold", "min": 1, "max": 20,
"step": 1, "default": 5 },
    { "type": "color", "id": "color", "label": "t:blocks.stock_badge.color", "default": "#b42318" }
  ]
```

```
}
{% endschema %}
```

`assets/stock-badge.css`:

```
.stock-badge { margin: 0.5rem 0; color: var(--stock-badge-color); font-weight: 600; }
```

STEP 02

The app embed

`blocks/help-bubble.liquid`: `"target": "body"` makes it an app embed, and `enabled_on` loads the script on product and collection pages only. With no link set, it renders nothing.

```
{%- if block.settings.link != blank -%}
  <aside class="help-bubble" data-help-bubble hidden>
    <a href="{{ block.settings.link }}">{{ block.settings.label | escape }}</a>
    <button type="button" data-help-close aria-label="{{ 'help_bubble.close' | t }}">&times;</button>
  </aside>
{%- endif -%}

{% schema %}
{
  "name": "t:blocks.help_bubble.name",
  "target": "body",
  "javascript": "help-bubble.js",
  "stylesheet": "help-bubble.css",
  "enabled_on": { "templates": ["product", "collection"] },
  "settings": [
    { "type": "url", "id": "link", "label": "t:blocks.help_bubble.link" },
    { "type": "text", "id": "label", "label": "t:blocks.help_bubble.label", "default": "Help" }
  ]
}
{% endschema %}
```

`assets/help-bubble.js` waits for the DOM, shows the bubble and remembers for the session that the shopper closed it, even when the browser blocks `sessionStorage`:

```
(() => {
  const KEY = "help-bubble-closed";
  const start = () => {
    const box = document.querySelector("[data-help-bubble]");
    if (!box) return;
    try {
      if (sessionStorage.getItem(KEY)) return;
    } catch {}
    box.hidden = false;
    box.querySelector("[data-help-close]").addEventListener("click", () => {
```

```
      box.hidden = true;
      try {
        sessionStorage.setItem(KEY, "1");
      } catch {}
    });
  };
  if (document.readyState === "loading") document.addEventListener("DOMContentLoaded", start);
  else start();
})();
```

`assets/help-bubble.css`:

```
.help-bubble { position: fixed; right: 1rem; bottom: 1rem; z-index: 10; display: flex; gap: 0.5rem; align-items: center; padding: 0.5rem 0.75rem; border-radius: 999px; background: #fff; box-shadow: 0 2px 8px rgb(0 0 0 / 0.15); }
.help-bubble[hidden] { display: none; }
.help-bubble button { min-width: 44px; min-height: 44px; border: 0; background: none; font-size: 1.25rem; cursor: pointer; }
```

STEP 03

Translations

Storefront text goes in `locales/en.default.json` and `locales/es.json`; editor labels go in `locales/en.default.schema.json` and `locales/es.schema.json`, with the same keys. The two English ones:

```
{
  "stock_badge": { "low_stock": "Only {{ quantity }} left in stock" },
  "help_bubble": { "close": "Close" }
}

{
  "blocks": {
    "stock_badge": { "name": "Stock badge", "product": "Product", "threshold": "Show at or below", "color": "Text color" },
    "help_bubble": { "name": "Help bubble", "link": "Help page", "label": "Text" }
  }
}
```

No comments and no trailing commas: extensions do not accept JSON with comments.

STEP 04

Validate, ship and retire script tags

1. `shopify theme check --path extensions/stock-badge` should report "no offenses found". Check the `target` by hand: a made-up value is not flagged.

2. `shopify app dev` opens the preview (Chrome only); `shopify app deploy` creates and releases a version. Limits: 10 MB, 30 blocks, 100 KB of Liquid.
3. Send the merchant into the editor with the embed on: `https://<store>.myshopify.com/admin/themes/current/editor?context=apps&template=product&activateAppId={client_id}/help-bubble`.
4. Confirm it is active with `app.extensions()` or in `config/settings_data.json`, and only then delete the script tags. Since October 1, 2026 none can be created; on March 1, 2027 they stop loading.

```
query AppScriptTags {
  scriptTags(first: 50) {
    nodes { id src displayScope createdAt }
  }
}

mutation DeleteScriptTag($id: ID!) {
  scriptTagDelete(id: $id) {
    deletedScriptTagId
    userErrors { field message }
  }
}
```

WHEN YOU FINISH

Theme cleanup checklist

For the code the app's old version left in the theme. Work on a copy.

- [ ] You duplicated the published theme and pulled the copy with `shopify theme pull`.
- [ ] You searched for the app and its script domain: `grep -rn "app-badge\|your-app.com" layout sections snippets templates`.
- [ ] You removed every `<script>` pasted into `layout/theme.liquid`, with no extra blank lines left.
- [ ] You removed each app snippet together with its `{% render %}`; never one without the other.
- [ ] `shopify theme check` passes on the cleaned copy.
- [ ] The page source no longer mentions the app's domain and the script loads once.
- [ ] The `scriptTags` query returns an empty list and the embed is active on the theme you are about to publish.
- [ ] Your app's onboarding screen carries these instructions and the deep link.